U.S. Patent Application

For:

SELF-CONTAINED APPLICATIONS ADAPTED TO BE RECEIVED BY AND PROCESSED WITHIN A BROWSER ENVIRONMENT

Inventor:
Daniel P. Veditz

97-870
013.0067

TITLE OF THE INVENTION

SELF CONTAINED APPLICATIONS ADAPTED
TO BE RECEIVED BY AND PROCESSED WITHIN
A BROWSER ENVIRONMENT

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to applications such as web-based applications (software systems) which are intended to be executed and manifested within a network client such as within a world wide web (WWW) browser environment.

Description of the Related Art

Network clients such as world wide web (WWW) enabled browsers are well known. In fact, browsers have become so widely used, that millions of people rely on the same to carry out their normal affairs. For example, consumers can now access the Internet to make travel arrangements, buy household items, and even trade securities such as stocks and bonds online. Also, users within organizations can access company web sites via internal networks commonly referred to as "intranets" to learn about company events, complete expense reports, exchange project information, etc. As a result, the browser has become almost a staple part of the way people use their computers to generate and receive information.

Although the browser network client, as a tool, has become very powerful and popular in terms of its inherent ability access server systems to receive content therefrom, the browser still remains limited in the functionality it can provide or allow. That is, browsers are designed to access remote server systems (computers) and receive content therefrom instead of providing

stand-alone functionality. In other words, although capable of loading files and processing the same from local storage facilities, they are targeted at accessing remote systems, receiving rich content, and providing a "current" web site experience. To date, browsers have not been utilized as general interfaces to facilitate general computing functionality that has traditionally been provided by stand-alone native applications and executable programs (e.g., form filling applications, etc.).

The aforementioned comments are not to be taken to indicate that developers have not created feature rich applications to operate within browser environments. To the contrary, many developers have created elaborate applications written as JAVA applications or "applets," for example, and have distributed the same for execution within a browser network client. Typically, however, such applets are executed within a processing space (e.g., within a window) inside of a browser environment to deliver a particular feature set. Unfortunately, such applets are also intended to be distributed from a server system via a network connection and merely form part of a larger web site environment which usually includes content of various types including, but not limited to, hyper-text content (i.e., hyper-text markup language (HTML) content), images, sound files, JAVA, JAVA Script, etc.

Despite the inherent ability of a browser to access server systems and receive and manifest feature rich content (e.g., such as that provided via JAVA, etc.), browsers continue to remain under-utilized. Such under-utilization also is due, in large part, to the fact that there is no current, effective and efficient way to package content and distribute the same via network connection, physical passage (e.g., via computer-readable media such as compact discs, etc.). There is no current way for developers to package all content that may be related to a particular web-based

application (e.g., one that would normally be served to a browser from a remote server system in the context of a web site, for example) and to distribute the same in an effective and efficient manner. For example, many people utilize open-standards based tools (e.g., HTML, etc.) to produce presentations, documentation sets, etc. which often include whole collections of files, images, etc. Save for successfully storing all such files in a commonly accessible directory, there is no way for a person to package all such files and make the same available or distribute such a package to client systems equipped with browsers. As such, there is no current way for a complete web-based application to be packaged for automatic processing within a browser environment and without the need for server (URL) distribution.

To address the aforementioned problems, developers have proposed several solutions which have not heretofore effectively allowed people to encapsulate complete web-based applications such as web sites. For example, many developers have developed file archiving formats to produce archive files containing all files related to a particular web application. Unfortunately, such currently available archiving schemes are not directed to the needs of a web-based application; that is, they are not directed to including the actual content necessary for a browser to render or layout web based data – e.g., HTML, etc. Moreover, such archiving schemes are not suited to produce auto-executable or auto-processable files within a browser environment –for example, there is no way to identify a starting file to cause a browser to instantly and automatically load and render a web-based application. And, despite their ability to encapsulate files, such currently available archiving schemes have no ability to provide a stand-alone file that may reside next to a native

application and which may be treated within an operating environment as one in the same.

Thus, there exists a need to allow web based application (e.g., those normally associated with a WWW site, etc.) to be encapsulated within a standard file structure, distributed via any form of data distribution, and which may be automatically executed within a browser environment without having to cause to the browser environment to access a remote server system for files forming part of the web based application. Accordingly, there exists a need to provide new and improved systems and methods for encapsulating a web-based application into an open-standards based file format and structure which may be processed by a correspondingly equipped network client or web browser. To be viable, such a system and method of encapsulation must be easy to use.

SUMMARY OF THE INVENTION

The present invention solves the aforementioned problems and, in so doing, provides certain benefits not heretofore realized with other network clients and software packaging schemes. In particular, the present invention provides a software component packaging scheme and corresponding network client that facilitates the receipt and processing of self-contained software systems (e.g., web sites, web-based applications, etc.) without requiring network access, etc. The present invention will allow developers and users to package whole collections of open-standards based content (e.g., all content associated with a web site, etc.) and to distribute the same to users who may access the same as though they were native, executable applications. Accordingly, the present invention extends the functionality of current-day web browsers beyond that of network content access

tool to allow the same to be considered as application processing environments capable of rich content manifestation, etc. Developers can utilize the present invention to deliver applications based on open-standards without having to be concerned with screen layout, content rendering, and other infrastructure issues which normally must be dealt with when developing stand-alone, native applications. Users will benefit from the present invention by becoming accustomed to a standard application processing environment in which many, if not all, of their applications are executed and run. And, users who are not otherwise connected or coupled to a network server (e.g., users of disconnected laptop computers, etc.) can now receive whole web-based applications and run the same like native applications without having to login into or access a remote server system.

The present invention solves the aforementioned problems and provides the above-described benefits by providing new and improved systems and methods for producing and processing self-contained software systems. In particular the present invention provides a self-contained software package that is adapted to be automatically processed within a browser environment. The self-contained software package includes, but is not limited to, a manifest containing meta-data about the self-contained software package, and at least one reference to an initial content source. The initial content source includes instructions related to a particular intended functionality such as that corresponding to a web site, open-standards application, etc. The manifest and reference(s) to the initial content source are adapted to be automatically received by and processed within a network client environment (e.g., within a world wide web – WWW browser). The initial content source is configured to control the

network client environment in accordance with the instructions to achieve the intended functionality.

According to another aspect of the present invention, provided is a web browser adapted to receive and process a self-contained software package. The web browser includes an input module for inputting a manifest containing meta-data about the self-contained software package, and at least one reference to an initial content source. The initial content source includes instructions related to a particular intended functionality such as that provided at or by a web site or by a web based/open standards application, etc. The manifest and reference(s) to the initial content source are automatically received by and processed within the web browser. The initial content source controls the web browser in accordance with the instructions to achieve the particular intended functionality.

According to another aspect of the present invention, provided is a method for generating and automatically processing a self-contained software package within a web browser environment. The method includes the steps of generating a manifest containing meta-data about the self-contained software package, generating at least one reference to an initial content source that includes instructions related to a particular intended functionality, automatically receiving the manifest, the reference(s), and the initial content source, and automatically processing the manifest, the reference(s), and the initial content source within the browser environment. The initial content source controls the browser environment in accordance with the instructions to achieve the particular intended functionality.

According to another aspect of the present invention, provided is a method of using a web browser that includes the steps of receiving a manifest containing meta-data about a self-

contained software package, receiving at least one reference to an initial content source that further includes instructions related to a particular intended functionality, and automatically processing the manifest, the reference(s), and the initial content source within a web browser environment. The initial content source controls the browser environment in accordance with the instructions to achieve the particular intended functionality.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is discussed in detail below with regard to the drawing figures attached hereto, of which:

FIG. 1 is a diagram of a system in which a client computing system (e.g., a personal data processing system such as a personal computer) is equipped with a network client (e.g., a world wide web browser software system) which is configured to receive and process a self-contained software package in accordance with a preferred embodiment of the present invention;

FIG. 2 is a block diagram of the client computing system depicted in FIG. 1;

FIG. 3 is a block diagram of a exemplary software system arranged and configured as a network client such as a web browser configured to be operated within the client computing system depicted in FIG. 2 and which is enabled to receive and process a self-contained software package in accordance with a preferred embodiment of the present invention;

FIG. 4 is a programmatic listing of a manifest file associated with a self-contained software package (e.g., archive, etc.) provided in accordance with a preferred embodiment of the present invention;

FIG. 5 is a programmatic listing of a manifest file corresponding to an object referenced in the manifest file depicted in FIG. 4; and

FIG. 6 is a flowchart that illustrates process steps which are performed to allow a network client such as a web browser to receive and process a self-contained software package in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention is now discussed in detail with regard to the drawing figures that were briefly described above. Unless otherwise indicated, like parts and processes are referred to with like reference numerals.

Referring now to FIG. 1, depicted therein is a diagram of a system 100 that includes the Internet 104 and/or other networks and wherein a client system equipped with a network client software package (e.g., a world-wide-web browser, etc.) may receive and process self-contained software packages in accordance with the present invention. In particular, system 100 includes a web server system 102 such as one equipped with a web server software system similar or like the NETSCAPE ENTERPRISE SERVER which is manufactured and marketed by NETSCAPE COMMUNICATIONS, INC. System 100 also includes the Internet 104 or other networks such as an intranet, etc. In the context of the present invention, a network is a collection of automatic data processing systems which are coupled together via data communications links. System 100 further includes an ISP 106 (Internet Service Provider) that is capable of allowing user systems to access Internet 104 via modem facilities, ISDN facilities, leased data communications

links, etc. Coupled to ISP 106 is client system 108 which is equipped, in accordance with the present invention, with a browser software system which is configured to receive self-contained software packages.

It should be noted that there is no requirement that ISP 108 be present in the context of the present invention. For example, if network 104 is an Intranet or other private/self-contained network, there is no need to provide for network access to outside users, etc. that may otherwise be remotely connected to network resources and who may require dial-up connections, etc.

In system 100, a software distributor 110 is shown as being coupled to client system 108 via a dashed line. The dashed line in system 100 is meant to indicate that the coupling thereof may be an electronic data communications coupling, or one that allows a client system owner and/operator to acquire an executable, encapsulated software application (e.g., web-based application, etc.) in accordance with the present invention such as via other means such as through retail channels, etc.

Referring now to FIG. 2, depicted therein is an exemplary arrangement of client system 108 as originally depicted in FIG. 1. In particular, client system 108 is a personal automatic data processing system such as one similar or like a personal computer manufactured and marketed by IBM CORPORATION and which includes a processor arrangement 202 of one or more processing elements, a data storage subsystem 204, and an I/O subsystem 206 to allow network communications, etc. Data storage subsystem 204 may be configured with a disk array to allow software packages to be loaded thereon and loaded into appropriate memory systems for execution by processor arrangement 202. Furthermore, client system 108 is configured to operate, in accordance with the present invention, a network client

such as a world wide web (WWW) browser software package that is further configured to receive encapsulated, self-contained executable software modules which may control the browser environment.

In regard to FIGS. 1 and 2, a self-contained executable software package that may be distributed to client system 108 has been developed. Such a self-contained, executable software package may be used to wrap up web pages, scripts, images, sound files, JAVA, etc. into a self-contained unit for processing within a browser software environment in accordance with the present invention. With such a self-contained unit, one could e-mail the executable package (or a link to it, etc.) to people and it would run automatically within a browser environment as provided by the present invention. An enterprise technology department could update the self-contained unit onto all of their employee's machines, for example, such as for mission critical internal web applications thus saving repeated downloads and resources. Moreover, a sales person could take a slide presentation on the road, or distribute the same without having to worry about losing pages or images during other archiving processes. Sales people could even pack an entire web site or Visual JAVAscript project, etc., and take it on the road. The same would run without having to install a server software system to run the same on a laptop computer, for example. Thus, in order to make more powerful web based applications, the present invention provides a convenient way for Visual JAVAscript and tools like it or similar to it to create a single file that is self-contained with everything needed to run a web based application (e.g., a web site). Accordingly, the present invention provides what may be referred to as a .XJR file. An .XJR file, in accordance with the present invention, uses the standard .JAR file format as proposed by

NETSCAPE COMMUNICATIONS INC. and SUN MICROSYSTEMS, INC., with the extension change from .JAR to .XJR for "executable .JAR". The .XJR file format allows for a new mime type (e.g., like or similar to "applications/crossware-app", etc.). The .XJR file, however, has some special headers in its manifest file. Manifest files will be immediately understood by those skilled in the art as they relate to .JAR type files. For example, the initial page or file (e.g., HTML file) to use when loading an encapsulated web application into an equipped browser is identified, in accordance with the present invention, in a manifest file associated with an archive containing all content necessary for a particular web application. The content (e.g., initial content—HTML—may also be included directly in the .XJR provided by the present invention). Additionally, an .XJR file manifest contains a new archive type identifier such as "WebApp" to indicate that the associated archive (.XJR archive) is a web based application such as one that corresponds to a web site, etc. (e.g., one having HTML and other types of content).

The creation of an .XJR file in accordance with the present invention will be similar to the creation of .JAR type files and archives as is well known in the art. It is the present invention's addition of initial content identifiers, etc. (e.g., the inclusion of an initial content source like or similar to an initial HTML file) that allows .XJR files to be created and processed by a browser (web browser, etc.) that is equipped to handle and process the same.

Referring now to FIG. 3, depicted therein is a block diagram of a software system arranged and configured as a network client such as a WWW browser software system. In particular, network client 300 includes I/O modules to facilitate input and output of content and other data, a processing engine 304 to facilitate control of the other software modules and

components within network client 300, and a rendering and layout engine to facilitate multimedia output such as screen output, sound output, printed output, etc. It is important to note that network client 300 and the block diagram depicted in FIG. 3 illustrates exemplary structures which would be found within a browser environment and, according to the present invention, one that is configured to receive self-contained software packages which may be loaded into processing engine 304 to facilitate a feature set related to a particular web based application. Such a self-contained application will come in the form of an .XJR file or equivalent thereof received via a network connection, a computer readable media such as a compact disc, etc. Processing engine 304 is configured with modules and objects to facilitate processing of .XJR and equivalent files in accordance with the present invention.

Accordingly, with reference to FIG. 4, depicted therein is a manifest file 400 which is associated with an .XJR file and a related archive that contains all content related to a self-contained web application (as indicated at the new "archive type" identifier referred to as "WebApp") in accordance with the present invention. Manifest file 400 includes a reference to an object 402 known as "index.html" which contains HTML content corresponding to a web based application as provided by the present invention. Accordingly, the present invention's use of a .JAR type file structure (archive) and manifest file to allow identification of web application components now allows web developers to facilitate the distribution of self-contained, executable software applications (web sites, web applications, etc.) to users to run the same within appropriately equipped web browser environments such as one similar or like that which is depicted in FIG. 3. By using a standard file structure like the .JAR

file structure which has now been renamed in accordance with the present invention as an .XJR for "executable .JAR" file, web developers may now distribute open-standards based archives containing whole web applications without the need for server systems that distribute web content and the like. For example, a presentation created using multiple HTML pages, may now be packaged within an archive associated with an appropriate manifest file and distributed via a network, via retail delivery, etc. to users for automatic execution within appropriately equipped browser environments.

Referring now to FIG. 5, depicted therein is a manifest file associated with a .JAR file component as identified within manifest 400. Manifest file 504 is a manifest file for the component "mycomponent.jar" which is a JAVA archive. Accordingly, the present invention allows even .JAR files and other type of archive files to be included within an archive containing all content necessary for a web site, or other web application. Accordingly, FIG. 5 clearly illustrates the recursive nature of the present invention to allow self-contained units to contain content formatted in accordance with a standard file format (i.e., .XJRs within .XJRs, etc.).

The systems illustrated in FIG. 1, 2, and 3 and the file formats created in accordance with the present invention as illustrated in FIGS. 4 and 5, are designed to operate together to facilitate the distribution of self-contained web based applications or that content which is executable and processable within a network client environment such as within a WWW browser environment. Accordingly, FIG. 6 illustrates a flowchart that depicts process steps that may be carried out to receive such a self-contained software package and to have the same automatically manifest as a web application within a browser

environment. In particular, such process steps commence at step S6-1 and immediately proceeds to step S6-2.

At step S6-2, a self-contained software package in the form of an .XJR file as discussed above with regard to FIGS. 4 and 5, will be received by a client system such as client system 108. Such receipt may be via a network coupling, other data communications coupling, or via distribution via some form of computer readable medium such as via retail distribution of a compact disc, etc.

Next, at step S6-3, if not already done so within a processing environment such as within an operating system (e.g., within the MICROSOFT WINDOWS 95™ or 98™ operating system environments, etc.), a browser or other network client may be spawned and loaded with the self-contained software package (i.e., the .XJR file as already received at step S6-2). The browser environment will then proceed to access the manifest file (e.g., manifest file 400 as depicted in FIG. 4) and proceed to locate all files within an associated archive and to process the same to provide functionality in accordance with instructions and content contained within files and data in the associated archive. That is, the manifest file of the .XJR file may be accessed for a start up file known as "index.html", which contains HTML instructions for controlling a browser environment and rendering or otherwise laying out content in accordance with instructions contained therein. Such instructions may call for the layout and rendering of content including images, text, sound files, etc. All types of content that may be displayed and/or manifested within a web site or web application are contemplated by the present invention.

Next, processing proceeds to step S6-4 where the browser environment accesses an extracts objects from the .XJR file and

executes accordingly as discussed above based upon instructions contained within content thereof etc.

Processing ends at step S6-5.

Based on the forgoing discussion, the present invention will allow an entire web application to be packaged as a modern equivalent of an .EXE (executable) file. With the use of an .XJR type file, as provided by the present invention, a browser environment may be spawned and caused to open and access content within a manifest file and an associated archive containing web site, web applications, etc. and other content for display and manifestation by the web browser. Alternatively, one could associate .XJR files in a WINDOWS type desktop so that they can automatically launch a browser upon instantiation thereof (e.g., via double clicking on an .XJR file icon in accordance with the present invention). Accordingly, applications may be built to run in a browser environment so that they can compete more directly with native machine applications, thus enhancing a browser platform configured and equipped in accordance with the present invention. Accordingly, a browser environment may now be caused to become more of a staple tool much like a word processor, spreadsheet, etc.

Furthermore, because an .XJR file according to the present invention bears the same file structure as a .JAR type file as will be readily understood by those skilled in the art, the same may be signed, thus establishing security credentials quickly and easily across all files within an .XJR file with a single signature thus greatly streamlining the need to sign individual scripts, and other components contained therein.

As a result, the present invention makes a browser type environment a more powerful place to deploy more powerful web applications, such as those built with Visual JAVAscript, etc.

Thus, having fully described the present invention by way of example with reference to the attached drawings figures, it will be readily appreciated that many changes and modifications may be made to the invention and to any of the exemplary embodiments shown and/or described herein without departing from the spirit or scope of the invention, which is defined in the appended claims.